# Advanced Topics and Post-relational Databases

- High-level data modelling
  - Enhanced ER Modelling
  - Object-oriented Modelling
- Indexes
- Real SQL Programming
- Data Warehousing and Mining

Readings: Sections 4.1.11, 4.6, 4.9, 8.3, 8.4, 9.1and 9.2 of textbook.

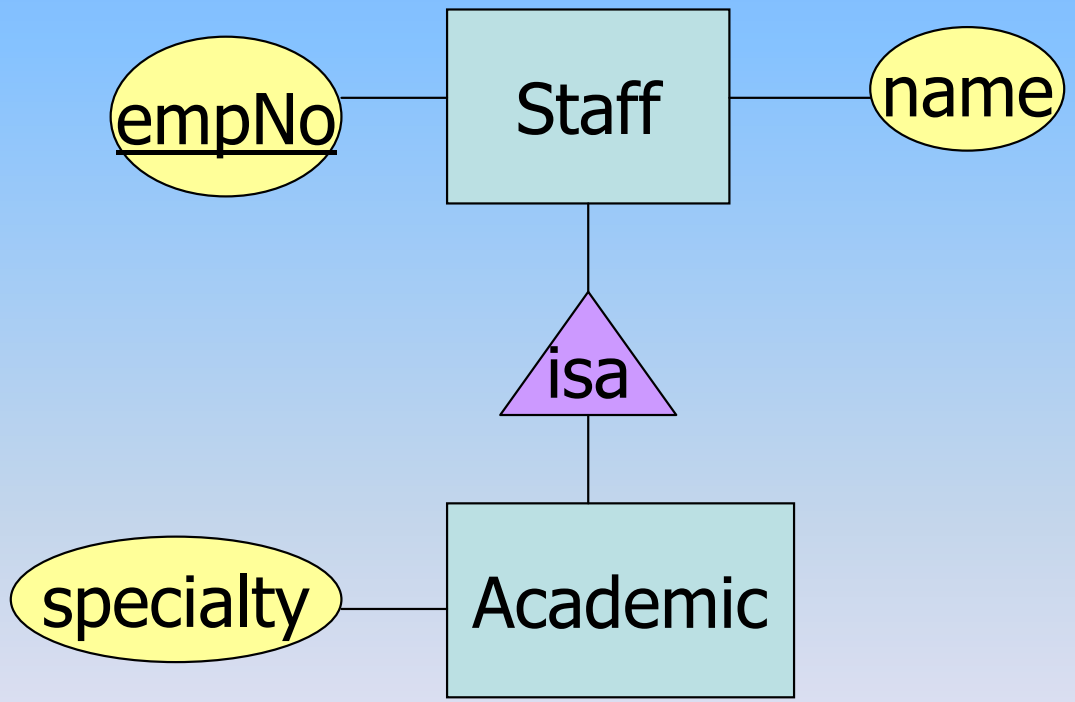
# Enhanced ER Modelling: Subclasses

- Subclass = special case = fewer entities = more properties.
- Example: Staff--Academic
  - Not every Staff member is an Academic, but some are.
  - Suppose that in addition to all the attributes and relationships of Staff, Academics also have the attribute specialty-area.

# Subclasses in E/R Diagrams

- Assume subclasses form a tree.
  - I.e., no multiple inheritance.
- Isa triangles indicate the subclass relationship.
  - Point to the superclass.

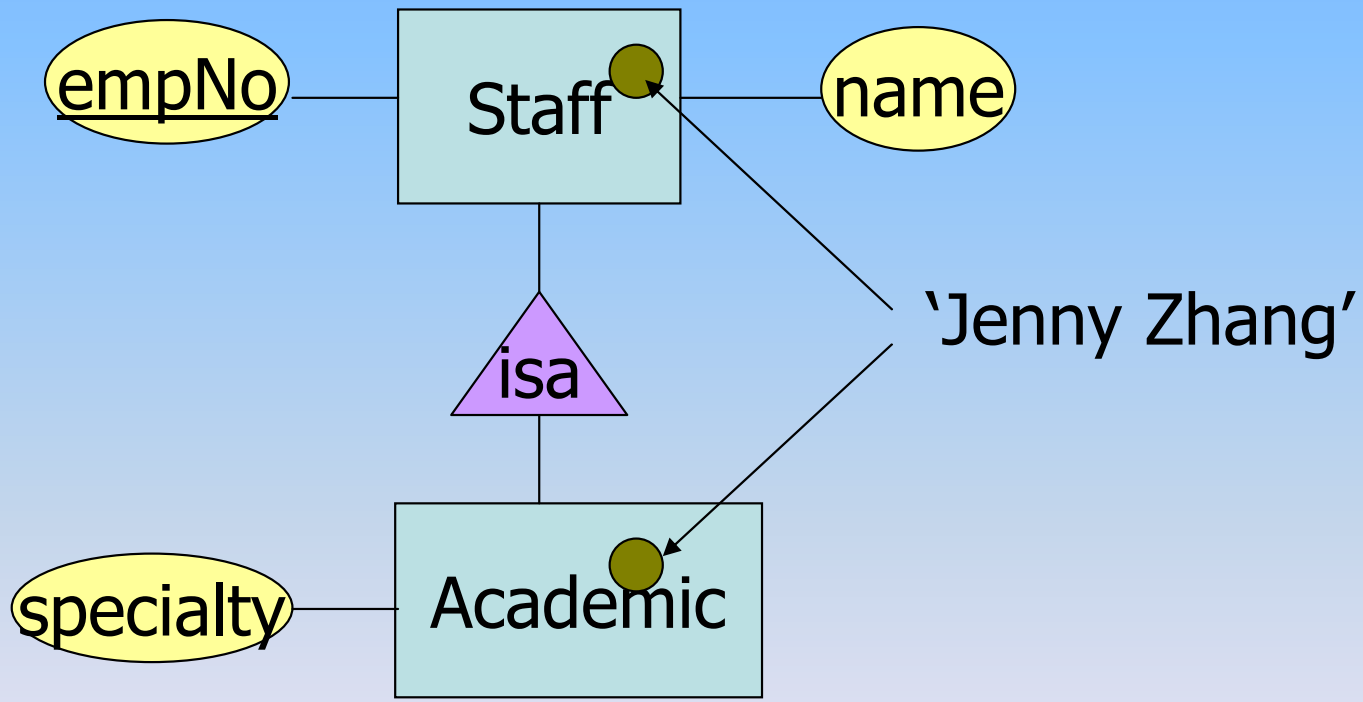
# Example: Subclasses



# E/R Vs. Object-Oriented Subclasses

- In OO (to be described in more details), objects are in one class only.
  - Subclasses inherit from superclasses.
- In contrast, E/R entities have representatives in all subclasses to which they belong.
  - Rule: if entity *e* is represented in a subclass, then *e* is represented in the superclass (and recursively up the tree).

# Example: Representatives of Entities



# Subclass to Relations – Object-Oriented Style

**Staff**

| empNo | name |
|---|---|
| e10000 | Peter Smith |

**Academic**

| empNo | name | specialty |
|---|---|---|
| e48154 | Jenny Zhang | Databases |

- Good for queries like "find the academics whose specialty is Databases".

# Subclass to Relations – E/R Style

**Staff**

| empNo | name |
|---|---|
| e10000 | Peter Smith |
| e48154 | Jenny Zhang |

**Academic**

| empNo | specailty |
|---|---|
| e48154 | Databases |

Good for queries like
"List all staff (including academics)".

# Subclass to Relations – Using Nulls

**Staff**

| empNo | name | specialty |
|---|---|---|
| e10000 | Peter Smith | Null |
| e48154 | Jenny Zhang | Databases |

Saves space unless there are *lots* of attributes that are usually NULL.

# The Object-Oriented Concepts

- Every thing is an object. Objects are defined by static properties— attributes, and also dynamic properties – methods. Values for attributes can have structures, rather than being an atomic type. Values returned from methods are computed.
- Objects of the same kind form a class.
- A subclass inherits attributes and methods from its superclasses, and only explicitly declares attributes and methods specific to the subclass.

# Object-orientation: Example

```
Class Staff {
        attribute string empNo;
        attribute string name;
         attribute Struct Addr
                    {string street, string city} address;
        attribute date DOB;
        method integer Age() {calculate age from DOB}
};
Class Academic: Staff {
        attribute string specialty
    };
```

# Object-Oriented Databases

- ODL (Object Description Language), like CREATE TABLE part of SQL, defines persistent classes, whose objects are stored permanently in the database.
  - ODL classes look like Entity sets with binary relationships, plus methods.
- OQL (Object Query Language), simulate SQL in an OO paradigm.

# Indexes

- Index is a data structure used to speed access to tuples of a relation.
  - A DBMS uses an index on a table to search for a row rather than scanning the whole table. This greatly reduces search time and disk input/output.
- An index in databases is always balanced search tree with giant nodes (a full disk page) called a B-tree.
  - This topic will be discussed in more details in the course *Database Systems (COSC2406/2407).*

# Creating Indexes

- In Oracle (and most other DBMS's), implicit indexes are created automatically when the PRIMARY KEY constraint is defined.
- An index can also be created in Oracle by a CREATE INDEX statement.

```
CREATE INDEX indexname
        ON tablename (col1, col2, ...);
```

# Creating Indexes …

- An index named mvindx is created.

```
CREATE INDEX mvindx
ON Movie(rel_date);
```

- The USER_INDEXES table keeps indexes.

```
SELECT index_name, table_name
FROM user_indexes
WHERE table_name='Movie';
```

# Using Indexes

- When an index is created, a user does not need to open or use the index with a command. Rather an index is used by the DBMS for processing queries.
- Indexes are kept separately from their base tables.
- Every insertion or deletion in a table updates the index, which is added overhead on the system.

# Using Indexes …

```
select mvID, title, to_char(rel_date, 'DD-MON-YYYY')
from movie
where to_char(rel_date, 'MON-YYYY') = 'JUN-2009'
```

```
 MVID TITLE                    TO_CHAR(REL_DATE,
---------- ----------------- -----------------
         2 Coco Avant Chanel  25-JUN-2009
         4 The Proposal       18-JUN-2009
```

- The above query uses the index *mvindex* to get all the movies released in June 2009 more efficiently.

# Database Tuning

- A major problem in making a database run fast is deciding which indexes to create.
  - An index speeds up queries that can use it.
  - An index slows down all modifications on its relation because the index must be modified too.

# Database Tuning …

- Generally an index is created on a column if the column
  - is used very often in querying and joining,
  - has a big domain of values, or
  - contains many Null values.
    - Null values are removed in indexes.
- An index should not be created for
  - a very small table,
  - a column  not used often in queries, or
  - a table that often gets updated.

# SQL in Real Programs

- We have seen only how SQL is used at the generic query interface --- an environment where we sit at a terminal and ask queries of a database.
- Reality is almost always different: conventional programs interacting with SQL.

# Options

1. Code in a specialized language is stored in the database itself (e.g., SQL-PSM, PL/SQL).
2. SQL statements are embedded in a host language (e.g., C).
3. Connection tools are used to allow a conventional language to access a database (e.g., CLI, JDBC, PHP/DB).

# Stored Procedures

- PSM, or “persistent stored modules,” allows us to store procedures as database schema elements.
- PSM = a mixture of conventional statements (if, while, etc.) and SQL.
- Lets us do things we cannot do in SQL alone.

# Basic PSM Form

- CREATE PROCEDURE <name> (
  <parameter list> )
  <optional local declarations>
  <body>;
- Function alternative:

CREATE FUNCTION <name> (
  <parameter list> ) RETURNS <type>

# PL/SQL

- Oracle uses a variant of SQL/PSM which it calls PL/SQL.
- PL/SQL not only allows you to create and store procedures or functions, but it can be run from the generic query interface (sqlplus), like any SQL statement.
- Triggers are a part of PL/SQL.

# PL/SQL …

- In addition to stored procedures, one can write a PL/SQL statement that looks like the body of a procedure, but is executed once, like any SQL statement typed to the generic interface.
  - Oracle calls the generic interface “sqlplus.”
  - PL/SQL is really the “plus.”

# Form of PL/SQL Statements

```
DECLARE
   <declarations>
BEGIN
   <statements>
END;
.
run
```

- The DECLARE section is optional.

# Embedded SQL

- Key idea: A preprocessor turns SQL statements into procedure calls that fit with the surrounding host-language code.
- All embedded SQL statements begin with EXEC SQL, so the preprocessor can find them easily.

# Shared Variables

- To connect SQL and the host-language program, the two parts must share some variables.
- Declarations of shared variables are bracketed by:

  EXEC SQL BEGIN DECLARE SECTION;

  <host-language declarations>

  EXEC SQL END DECLARE SECTION;

# Use of Shared Variables

- In SQL, the shared variables must be preceded by a colon.
  - They may be used as constants provided by the host-language program.
  - They may get values from SQL statements and pass those values to the host-language program.
- In the host language, shared variables behave like any other variable.

# Host/SQL Interfaces Via Libraries

- The third approach to connecting databases to conventional languages is to use library calls.
  1. C + CLI
  2. Java + JDBC
  3. PHP + PEAR/DB

# Three-Tier Architecture

- A common environment for using a database has three tiers of processors:
  1. Web servers --- talk to the user.
  2. Application servers --- execute the business logic.
  3. Database servers --- get what the app servers need from the database.

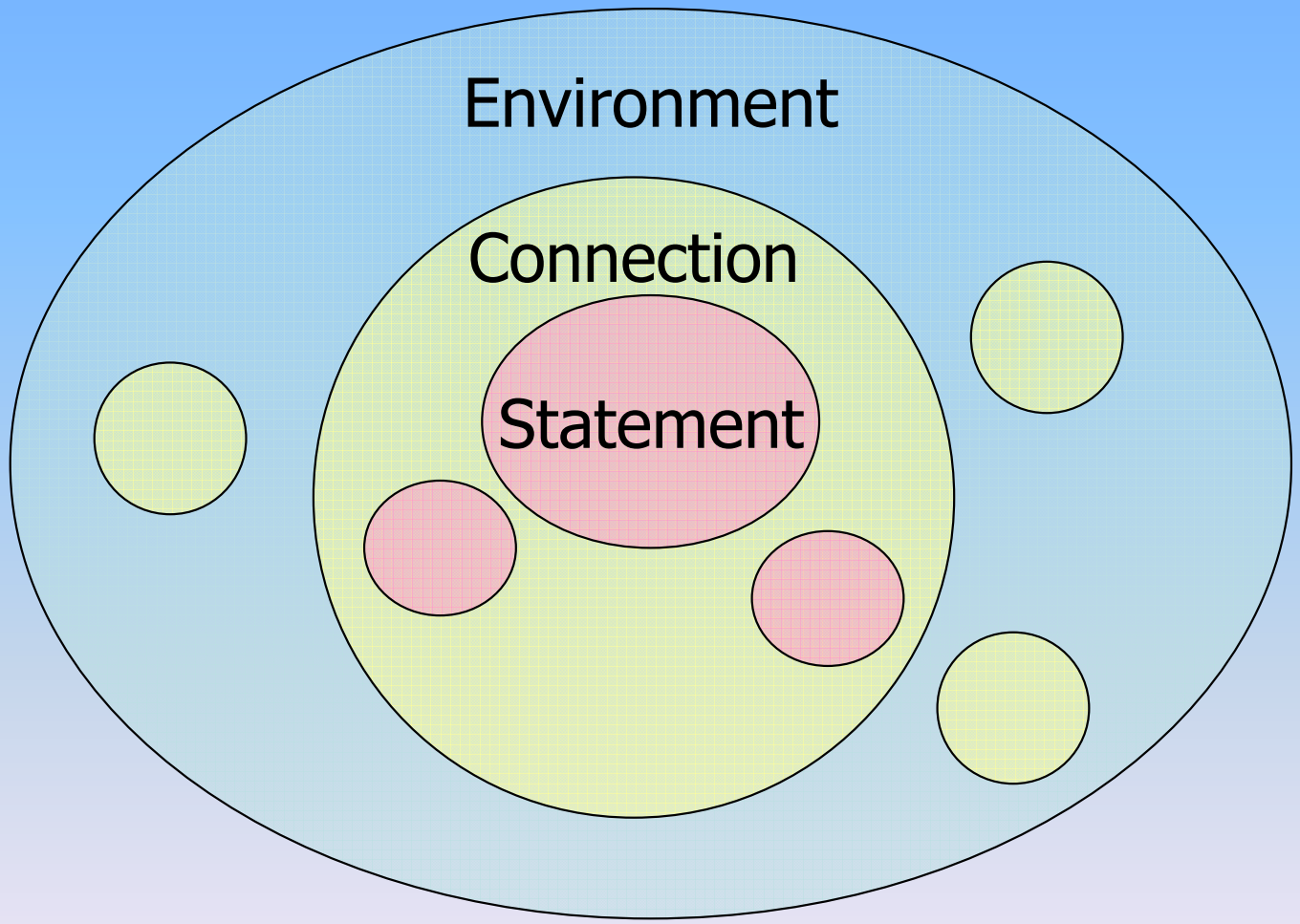
# Example: Amazon

- Database holds the information about products, customers, etc.
- Business logic includes things like “what do I do after someone clicks ‘checkout’?”
  - Answer: Show the “how will you pay for this?” screen.

# Environments, Connections, Queries

- The database is, in many DB-access languages, an environment.
- Database servers maintain some number of connections, so app servers can ask queries or perform modifications.
- The app server issues statements : queries and modifications, usually.

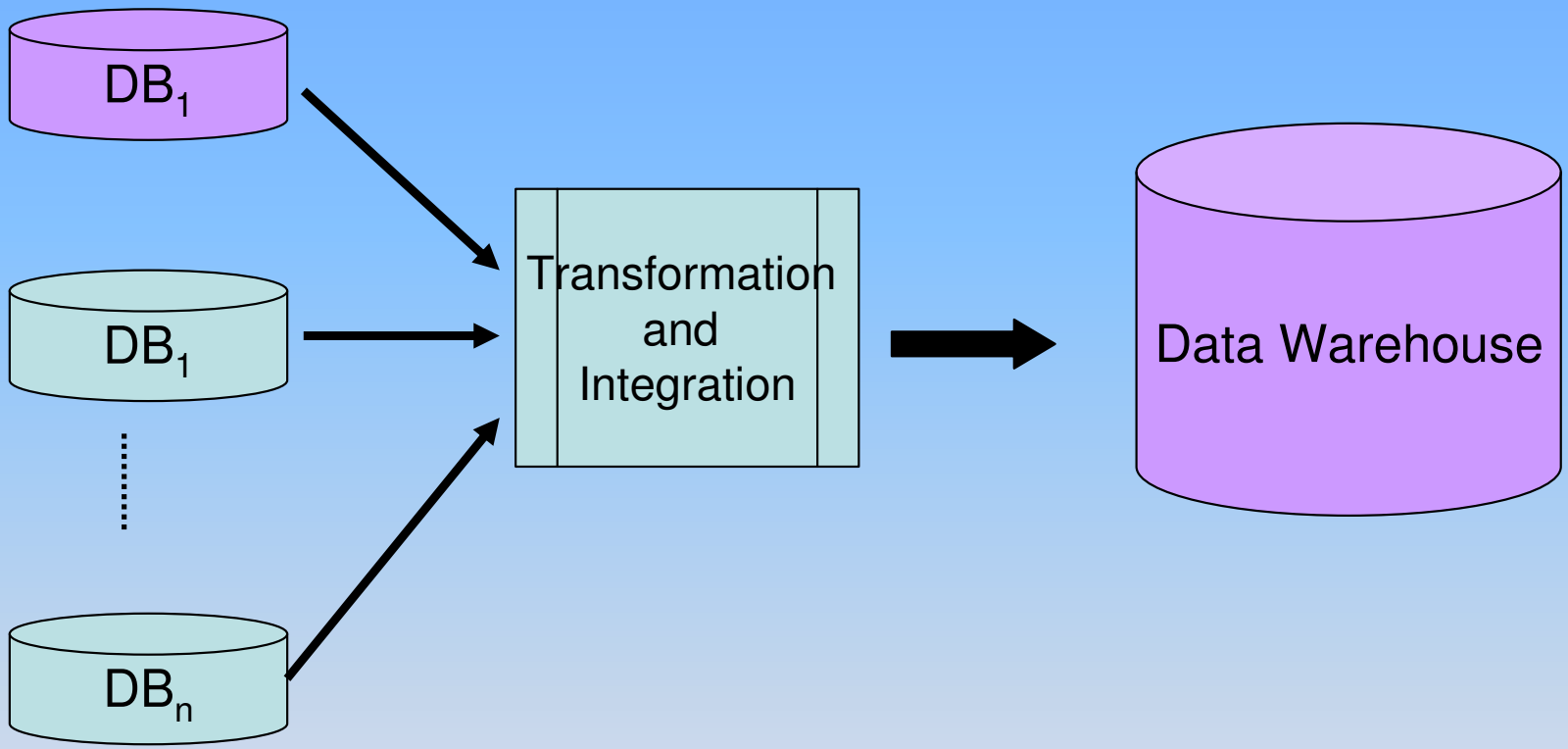
# Diagram to Remember



# Data Warehouses

- Enterprise Information Integration: making separate DB's, all owned by one company, work together.
- Existing databases may be incompatible in many ways.
  - *Lexical* : `addr` in one DB is `address` in another.
  - *Value mismatches* : is a "red" car the same color in each DB? Is 20 degrees Fahrenheit or Centigrade?
  - *Semantic* : are "employees" in each database the same? What about consultants? Retirees? Contractors?

# Data Warehouses …

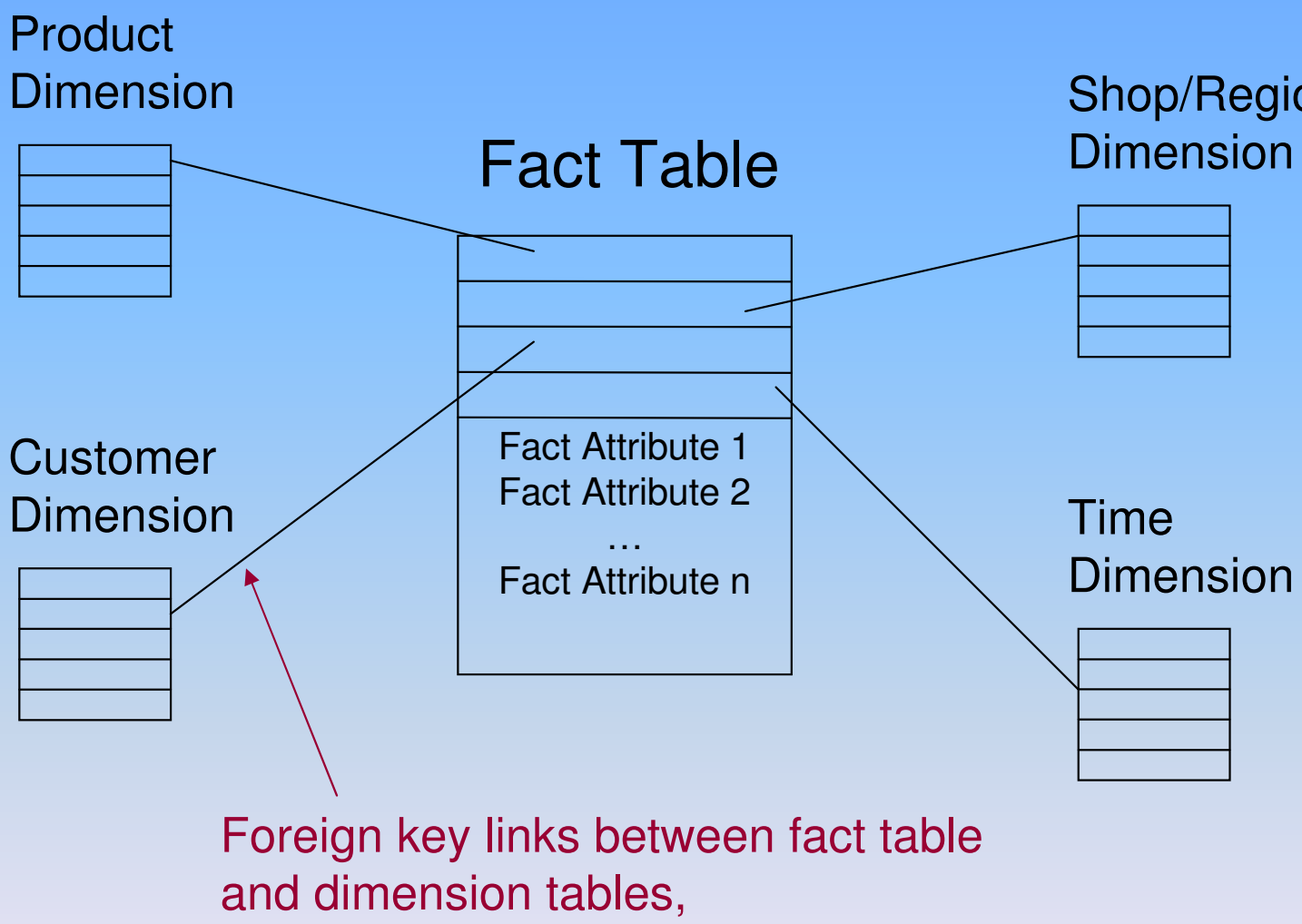



A data warehouse is a purposedly designed large repository of data.

# Data Warehouses: A Multi-dimensional View

- There are usually “aspects” (called dimensions) to aggregate and analyse facts. For example:
  - Dimensions: product, customer, region, time
  - Facts: dollar-sale, quantity-sale, expense, profit, loss

# Star Schema


Product
Dimension
Fact Table
Shop/Region
Dimension
Customer
Dimension
Fact Attribute 1
Fact Attribute 2
…
Fact Attribute n
Time
Dimension
Foreign key links between fact table
and dimension tables,

# Data Warehouses: OLAP

- Typically On-Line Application Processing (OLAP, or “analytic”) queries run on warehouses:
  - Few, but complex queries --- may run for hours.
  - Queries do not depend on having up-to-date data.

# OLAP: Example

- Typically OLAP is about aggregation and analysing trends.
  - Analysts at Microsoft would like to analyse the trend of sale of different products in Australia in the last 2 years.

# From Data Analysis to Data Mining

- Data mining is a popular term for queries that summarize big data sets in useful ways.
- Examples:
  1. Clustering all Web pages by topic.
  2. Finding characteristics of fraudulent credit-card use.

# Example: Mining Shopping Basket Data

- Mining for items that frequently appear in the same shopping baskets.
- If it is found that people often buy hamburger and ketchup together, the store can:
  1. Put hamburger and ketchup near each other and put potato chips between.
  2. Run a sale on hamburger and raise the price of ketchup.

# Advanced database courses

- Database indexing and query processing will be discussed in *Database Systems* (*COSC2406/2407*).
- SQL programming in a server environment will be discussed in *Implementation of Database Applications (ISYS1101/1102)* and *Web database Applications (ISYS1124/1126)*.
- The data warehousing technology and other topics will be discussed in more details in the course *Knowledge and Data Warehouses (ISYS1072/1073)*.
- The data mining technology will be discussed in *Data Mining (COSC2110/2111)*.

# Acknowledgements


- Some slides are based on the lecture slides by J.D. Ullman.